atp

atp当前提供的特性分为用户面特性和管理面特性。

- 用户面特性包含：选择测试场景、测试任务进展展示、测试报告分析、测试报告下载、贡献测试用例
- 管理面特性包含：测试场景管理、测试套管理、测试用例管理、测试模型一键导入、测试任务分析、测试任务管理、贡献管理

其中，用户面特性有以下接口：

- 1. Task
  - 1.1 POST create test task
  - 1.2 POST run test task
  - 1.3 GET get task list
  - 1.4 GET get one task
  - 1.5 PUT modify test case status
  - 1.6 POST batch delete test tasks
  - 1.7 GET get test tasks analysis

以执行测试任务接口1.2 POST run test task的实现：

```java
@Override
public TaskRequest runTask(String taskId, List<String> scenarioIdList) throws FileNotExistsException {
    //...param check
    //db操作，根据测试场景id scenarioIdList,拼装测试task结构
    //结构组成为： task(测试任务)-1/n->testScenario(场景)-1/n->testSuite(套件)-1/n->testCase(用例)
    initTestScenarios(scenarioIdList);
    TaskRequest task = taskRepository.findByTaskIdAndUserId(taskId, null);
    //...null&status check
    Map<String, String> context = AccessTokenFilter.CONTEXT.get();
    task.setTestScenarios(initTestScenarios(scenarioIdList));
    task.setAccessToken(context.get(Constant.ACCESS_TOKEN));
    task.setStatus(Constant.WAITING);
    taskRepository.update(task);
    String filePath = task.getPackagePath();
    //具体执行实现，调用线程池执行run()
    testCaseManager.executeTestCase(task, filePath);
    AccessTokenFilter.deleteContext();
    return task;
}
```

执行实现:

```
private class TaskProcessor implements Runnable {
 //...
 @Override
 public void run() {
 //更新任务状态
 task.setStatus(Constant.RUNNING);
 taskRepository.update(task);
 //填充测试需要的context,包括access token,tenantId,apm/appo/inventory/appStroe地址
 Map<String, String> context = new HashMap<String, String>();
  ...
  task.getTestScenarios().forEach(testScenario -> {
  //执行测试用例
  parseTestCase(testScenario, context);
  });
  task.setEndTime(taskRepository.getCurrentDate());
  task.setStatus(resultStatus);
  taskRepository.update(task);
  }
  private void parseTestCase(TaskTestScenario taskTestScenario, Map<String, String> context) {
  //逐层解析test task 至 测试用例列表
  }
  private void executeTestCase(List<TaskTestCase> taskTestCaseList, Map<String, String> context) {
  taskTestCaseList.forEach(taskTestCase -> {
  taskTestCase.setResult(Constant.RUNNING);
  taskRepository.update(task);
  // just execute automatic type test case
  if (Constant.TASK_TYPE_AUTOMATIC.equals(taskTestCase.getType())) {
  //获取db中的测试用例记录
  TestCase testCase = testCaseRepository
  .findByName(taskTestCase.getNameCh(), taskTestCase.getNameEn());
  setConfigParam(testCase, context);
  switch (testCase.getCodeLanguage()) {
  //根据语言类型执行测试脚本
  case Constant.PYTHON:
  PythonCallUtil.callPython(testCase, filePath, taskTestCase, context);
  break;
  case Constant.JAVA:
```

```java
JavaCompileUtil.executeJava(testCase, filePath, taskTestCase, context);
break;
case Constant.JAR:
JarCallUtil.executeJar(testCase, filePath, taskTestCase, context);
break;
default:
break;
}
if (!Constant.RUNNING.equals(resultStatus)) {
resultStatus = Constant.FAILED.equals(taskTestCase.getResult())
? Constant.FAILED
: resultStatus;
}
taskRepository.update(task);
} else {
// have manual test case, the total status is running
resultStatus = Constant.RUNNING;
}
});
}
```

以java为例，通过自定义类加载器，加载测试java程序，执行execute方法，其中测试程序位于项目目录atp/src/main/resources/testCase：

```java
public static void executeJava(TestCase testCase, String csarFilePath, TaskTestCase taskTestCase,
Map<String, String> context) {
try {
String className = testCase.getClassName();
Map<String, byte[]> bytes = compile(className.concat(Constant.DOT).concat(Constant.JAVA),
getFileContent(testCase.getFilePath()));
// put class into storage
try (JavaCompileUtil.MemoryLoader clsLoader = new JavaCompileUtil.MemoryLoader(bytes);) {
Class<?> clazz = clsLoader.loadClass(className);
 Object response = clazz.getMethod("execute", String.class, Map.class).invoke(clazz.newInstance(),
 csarFilePath, context);
 CommonUtil.setResult(response, taskTestCase);
 }

 } catch (Exception e) {
    ...
  }

}
```
管理面特性接口有，主要用于对测试用例、测试套件、测试场景进行CRUD操作，不再赘述：

- 2. Test case
    - 2.1 GET query all test cases
    - 2.2 POST create test case
    - 2.3 PUT update test case
    - 2.4 DELETE delete test case
    - 2.5 GET query one test case
    - 2.6 GET download test case
- 3. Test scenario
    - 3.1 GET query all test scenarios
    - 3.2 POST create test scenarios
    - 3.3 PUT update test scenarios
    - 3.4 DELETE delete test scenarios
    - 3.5 GET query one test case
    - 3.6 GET query all test cases under one scneario
- 4. Test suite
    - 4.1 GET query all test suites
    - 4.2 POST create test suite
    - 4.3 PUT update test suite
    - 4.4 DELETE delete test suite
    - 4.5 GET query one test suite
- 5. Contribution
    - 5.1 GET query all contributions
    - 5.2 POST create contribution
    - 5.3 POST batch delete contributions
    - 5.4 GET download contribution script
- 6. File